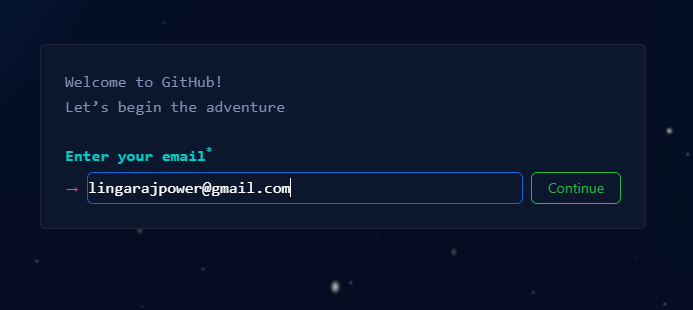
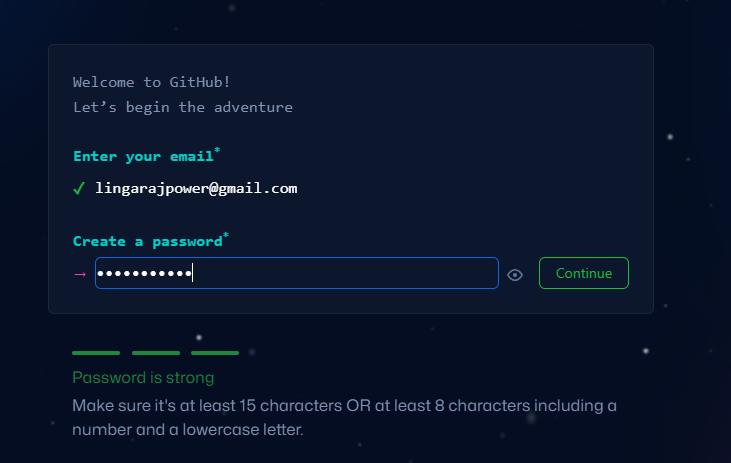
**Creating a account in Git Hub**

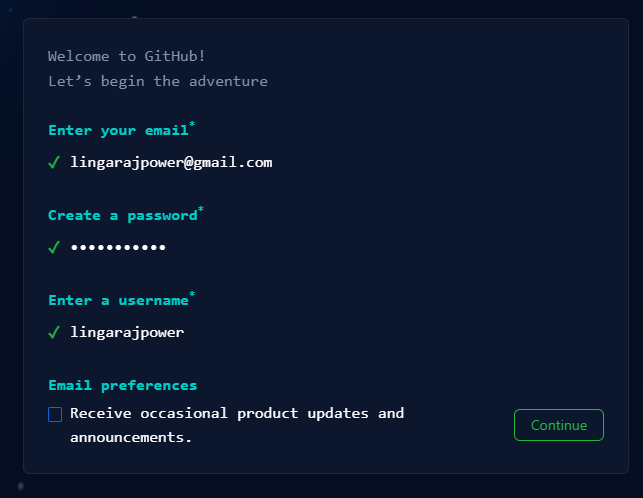
Step 1:



Step 2:

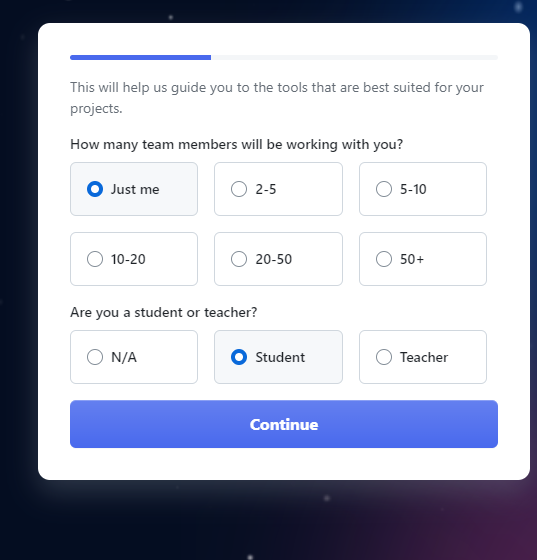


Step 3:



Step 4:

Verification in mail and OTP  
Step 5:

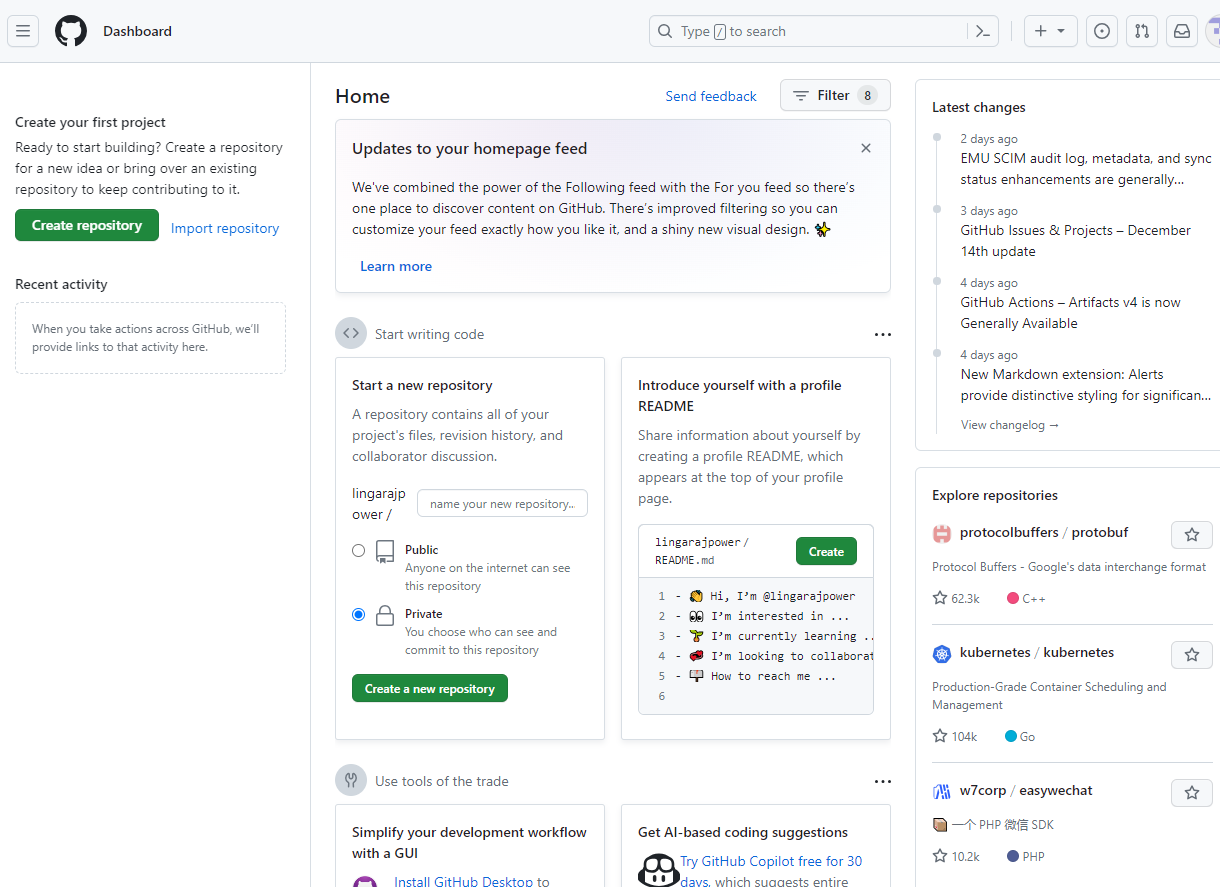


Step 7:

Select **Collaborative coding and community**

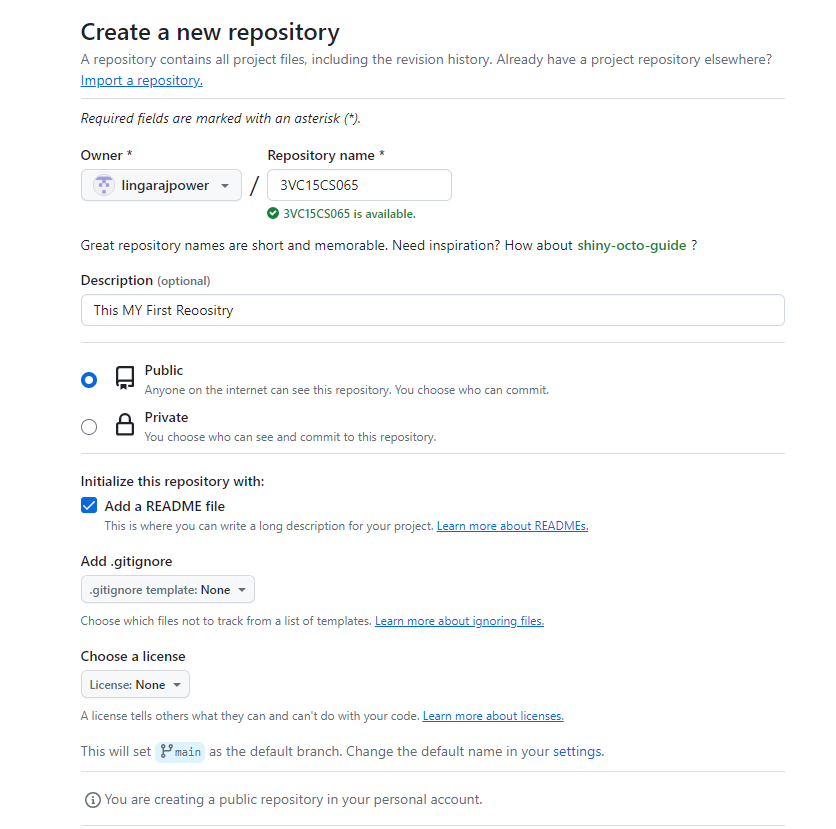
**Step 8: Select free plan.**

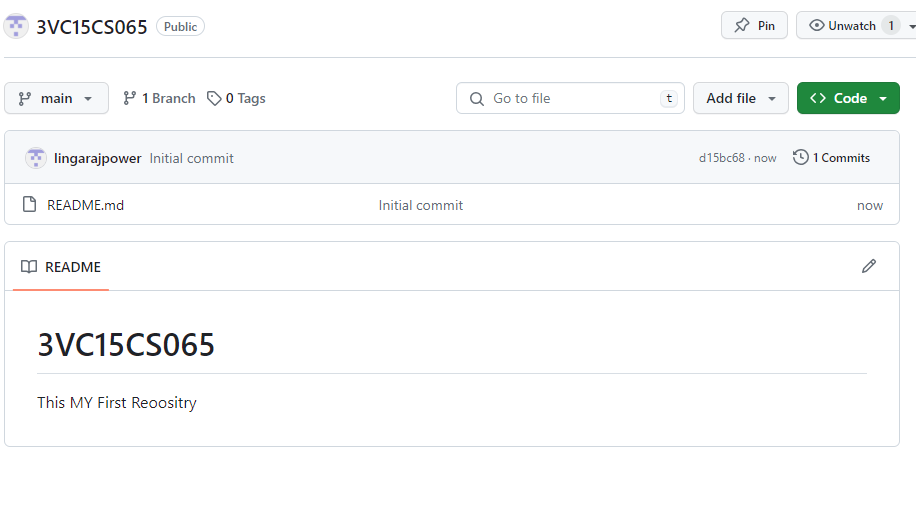
**Step 9:**



Step 10: Click your Profile 🡪 overview🡪 Repositories.

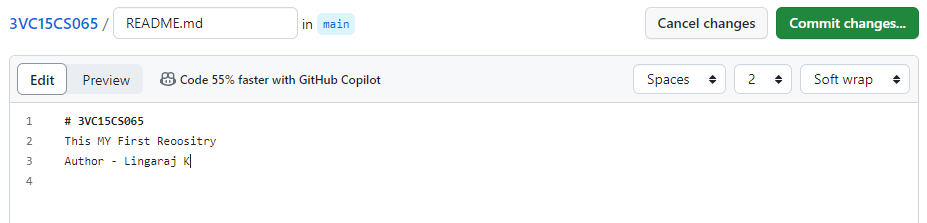
Step 11: Create repositories



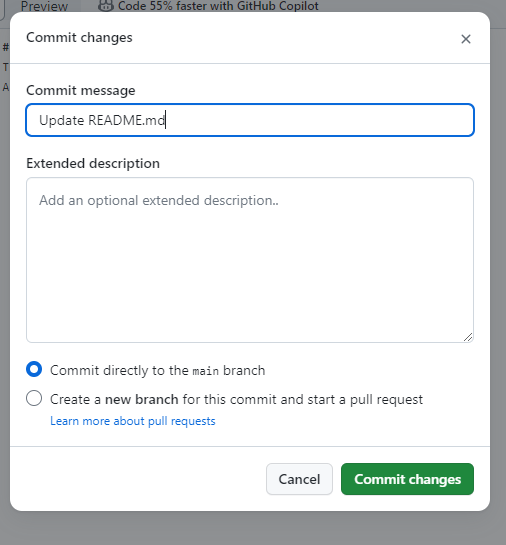


Step 12: Click Read MD file

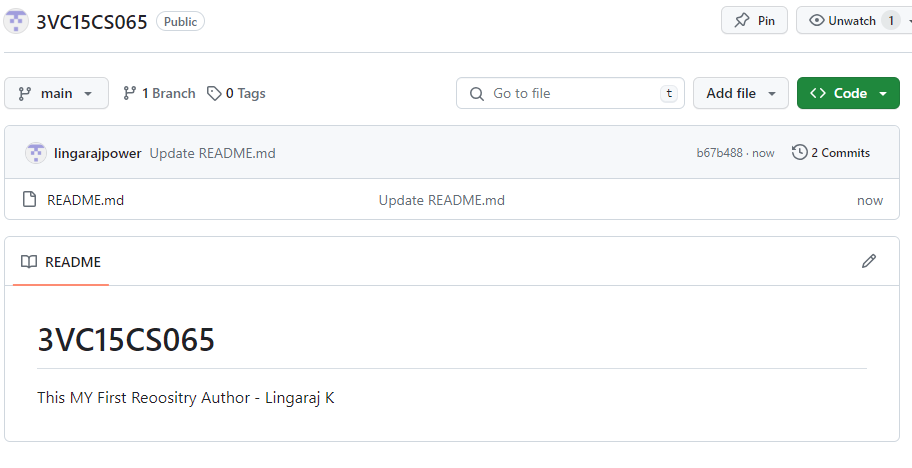
Step 13: to edit Read me file go to EDIT



Step 14: Click Commit Changes



Step 15: Cilck code



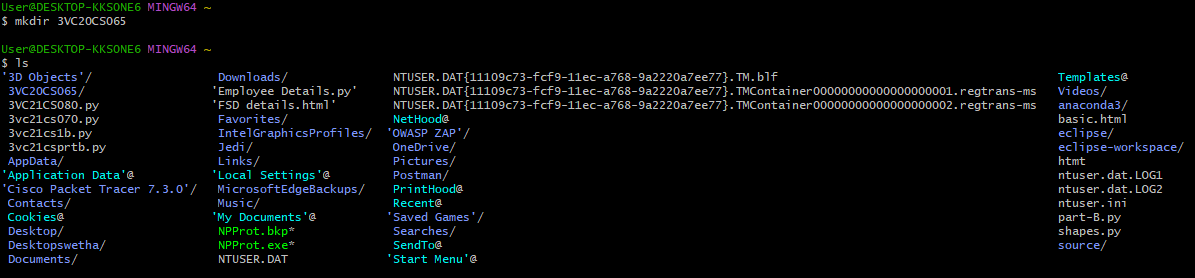
To code to better

<br> in code to get in next line.

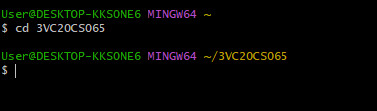
| **Sl. No** | **Experiments** |
| --- | --- |
| 1 | **Setting Up and Basic Commands**  Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message. |

Step1 : mkdir 3VC20CS065

Step 2: ls



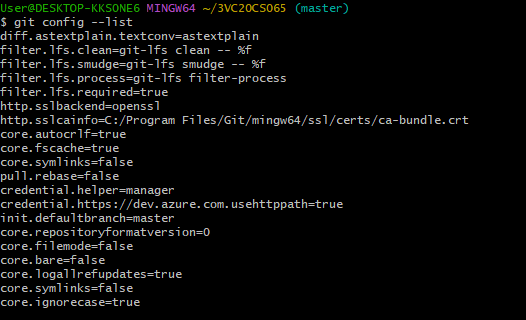
Step 3: cd 3VC20CS065



Step 4: To create a repository

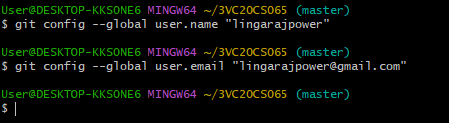
Git init

Git config-list

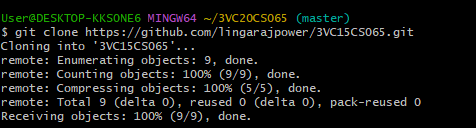


git config –global user.name “lingarajpower”

git config – global user.email “ lingarajpower@gmail.com”



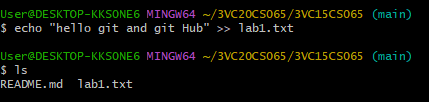
Git clone <https://github.com/lingarajpower/3VC15CS065.git>



cd 3VC15CS065

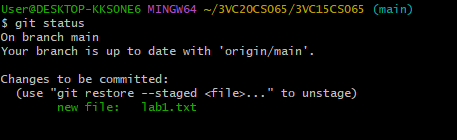
git status

$ echo "hello git and git Hub" >> lab1.txt

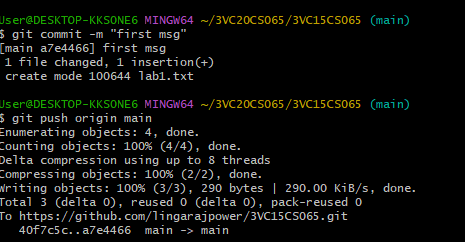


$ git add lab1.txt

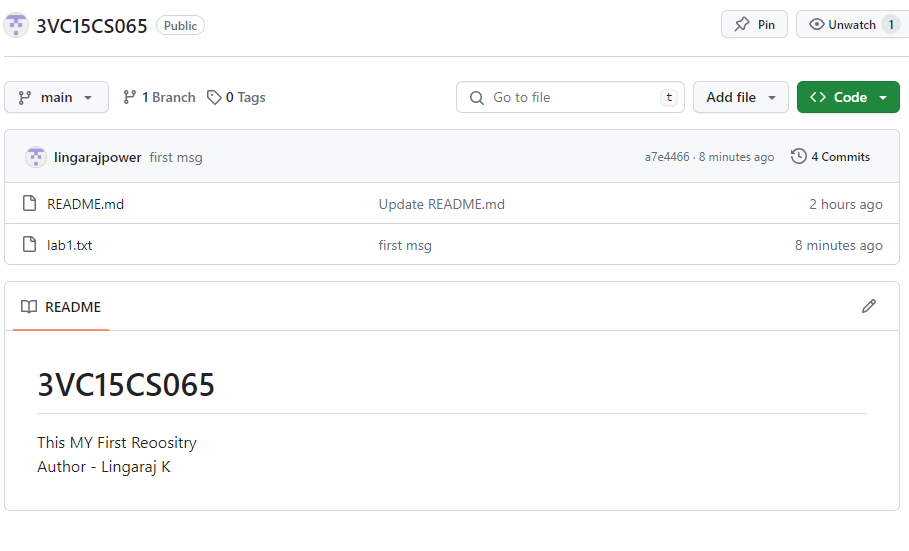
$ git status



Git commit –m “first msg”



Git push origin main



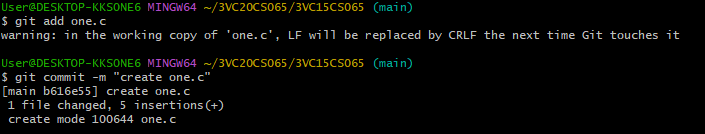
| **Sl. No** | **Experiments** |
| --- | --- |
| 2 | **Setting Up and Basic Commands**  Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message. |

$ vi one.c



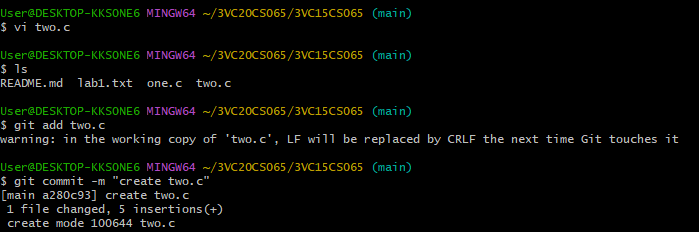
$ git add one.c

Git commit –m “create one.c”



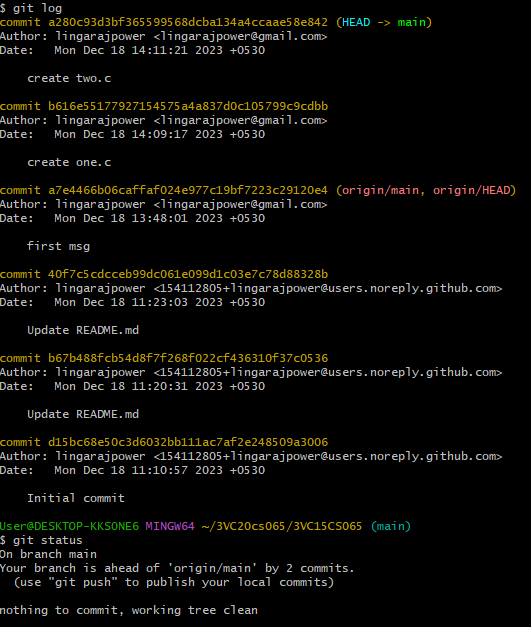
git add two.c

git commit -m "create two.c"



git log

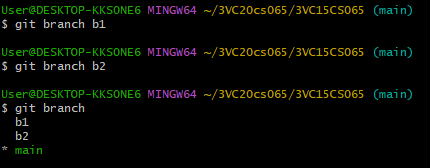
git status



$ git branch b1

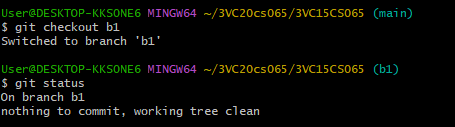
git branch b2

git branch



Git checkout b1

Git status



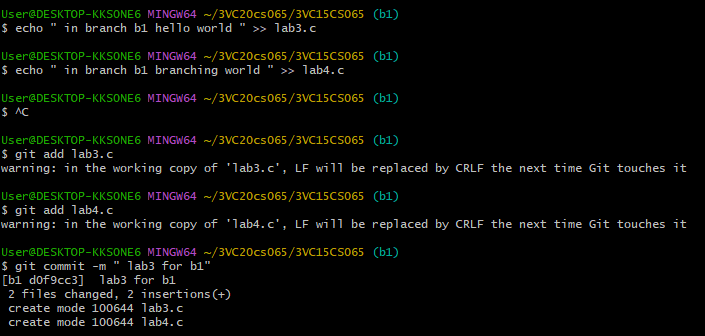
$ echo " in branch b1 hello world " >> lab3.c

$ echo " in branch b1 branching world " >> lab4.c

git add lab3.c

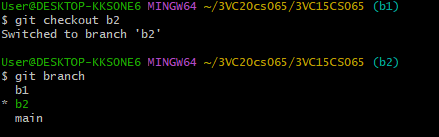
git add lab4.c

git commit -m " lab3 for b1"



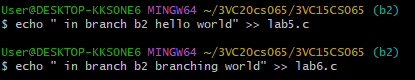
git checkout b2

git branch



echo " in branch b2 hello world" >> lab5.c

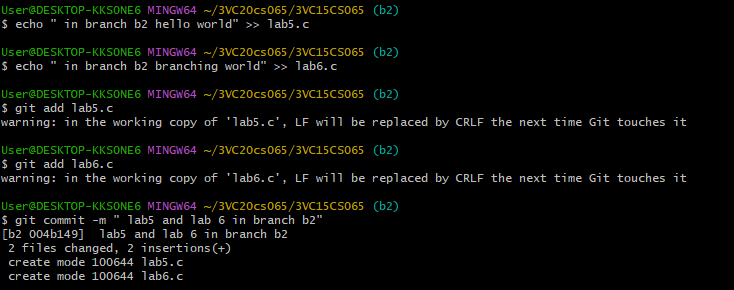
echo " in branch b2 branching world" >> lab6.c



git add lab5.c

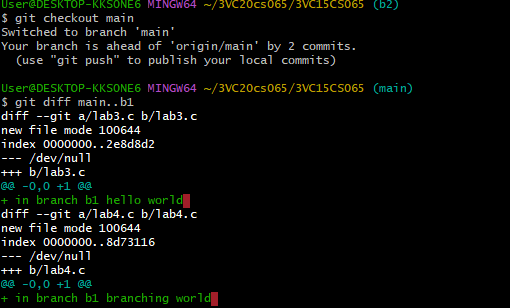
git add lab6.c

git commit -m " lab5 and lab 6 in branch b2"

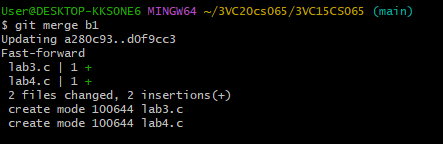


git checkout master

git diff main..b1



git merge b1

.

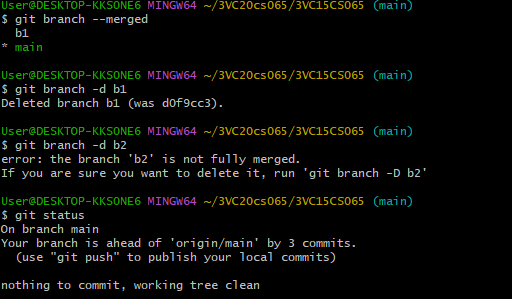
git branch –merged



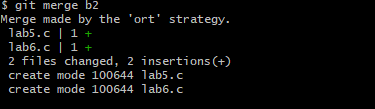
git branch -d b1

git branch -d b2

$ git status

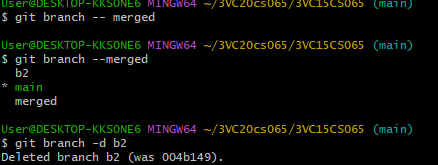


git merge b2



git branch –-merged

git branch -d b2

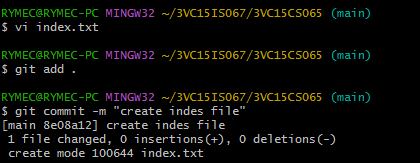


| **Sl. No** | **Experiments** |
| --- | --- |
| 3 | Creating and Managing Branches Write the commands to stash your changes, switch branches, and then apply the stashed changes |

vi index.txt

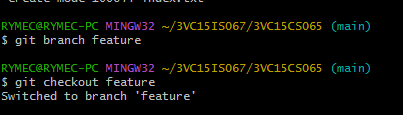
git add .

$ git commit -m "create indes file"



$ git branch feature

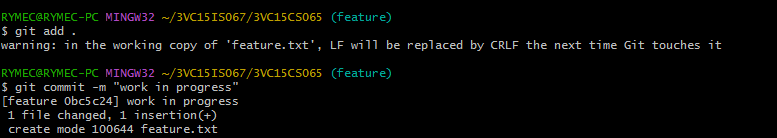
$ git checkout feature



vi feature.txt

git add.

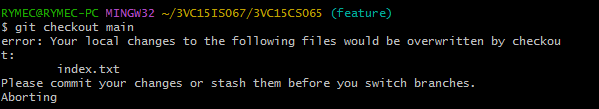
$ git commit -m "work in progress"



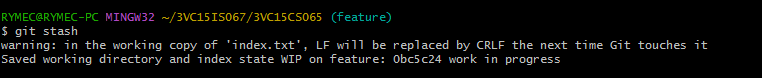
$ vi index.txt

$ git status

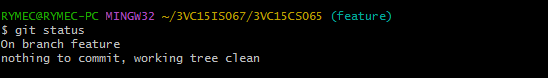
git checkout main



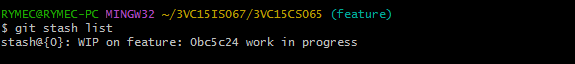
git stash



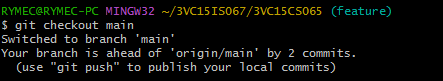
Git status



git stash list



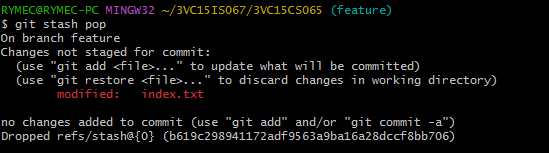
git checkout main



git checkout feature



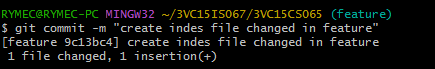
git stash pop



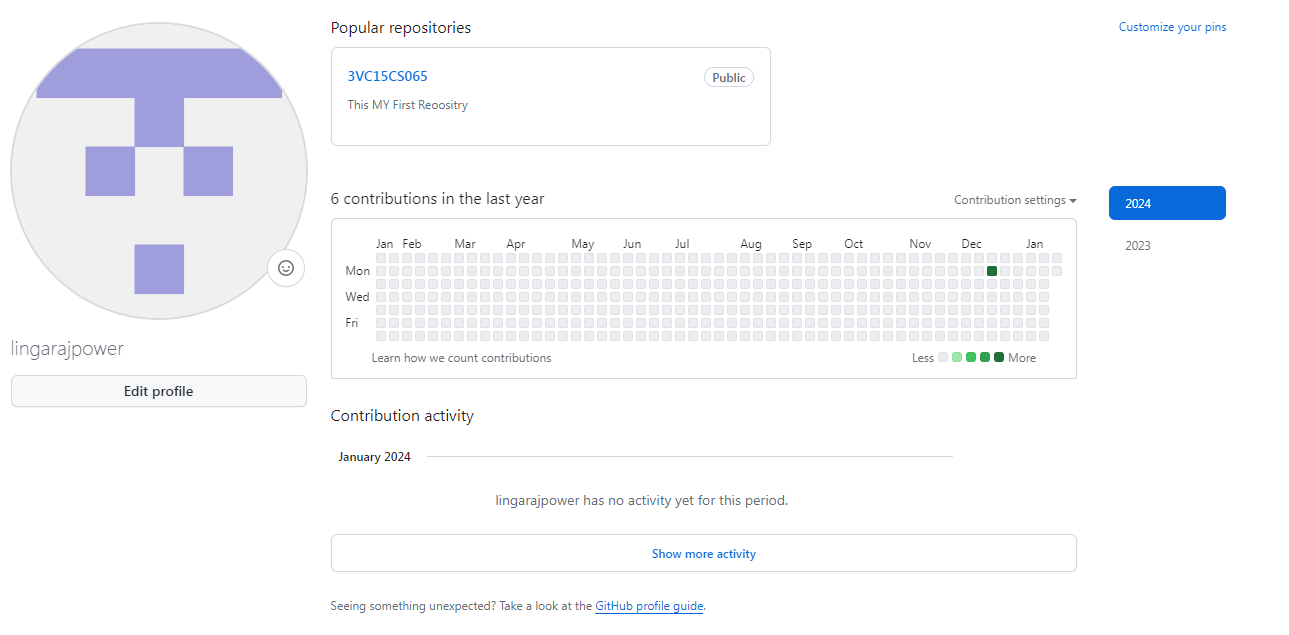
$ git stash list

$ git add .

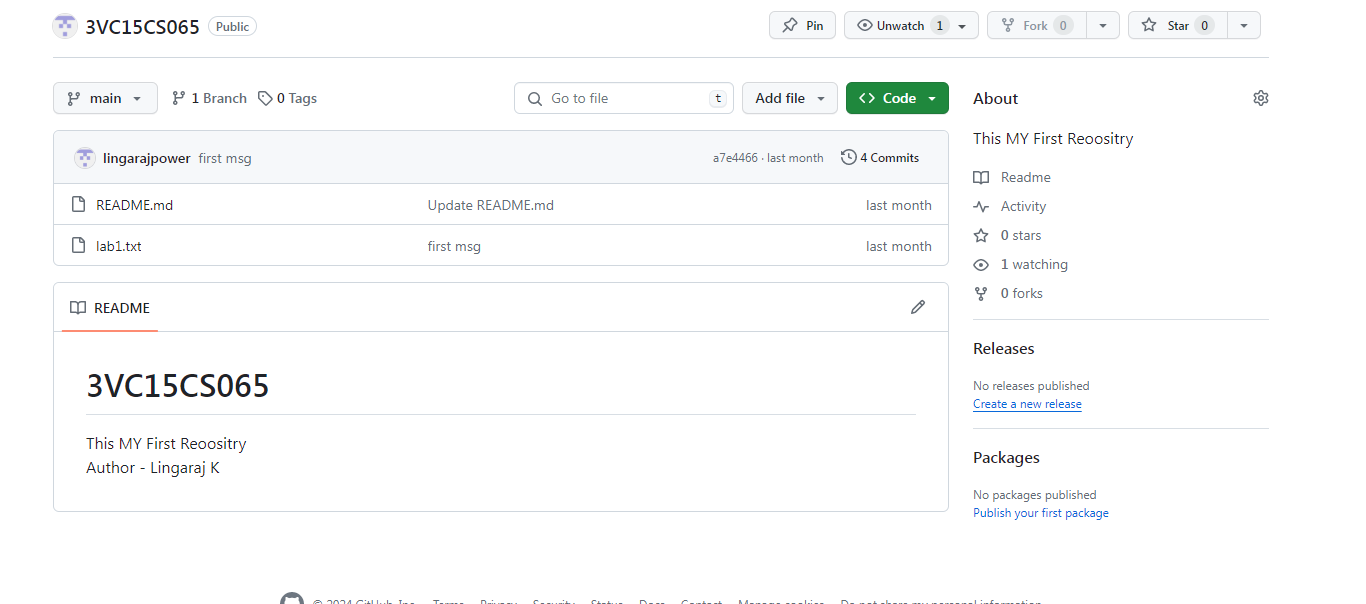
git commit -m "create indes file changed in feature"



| **Sl. No** | **Experiments** |
| --- | --- |
| 4 | Collaboration and Remote Repositories Clone a remote Git repository to your local machine |

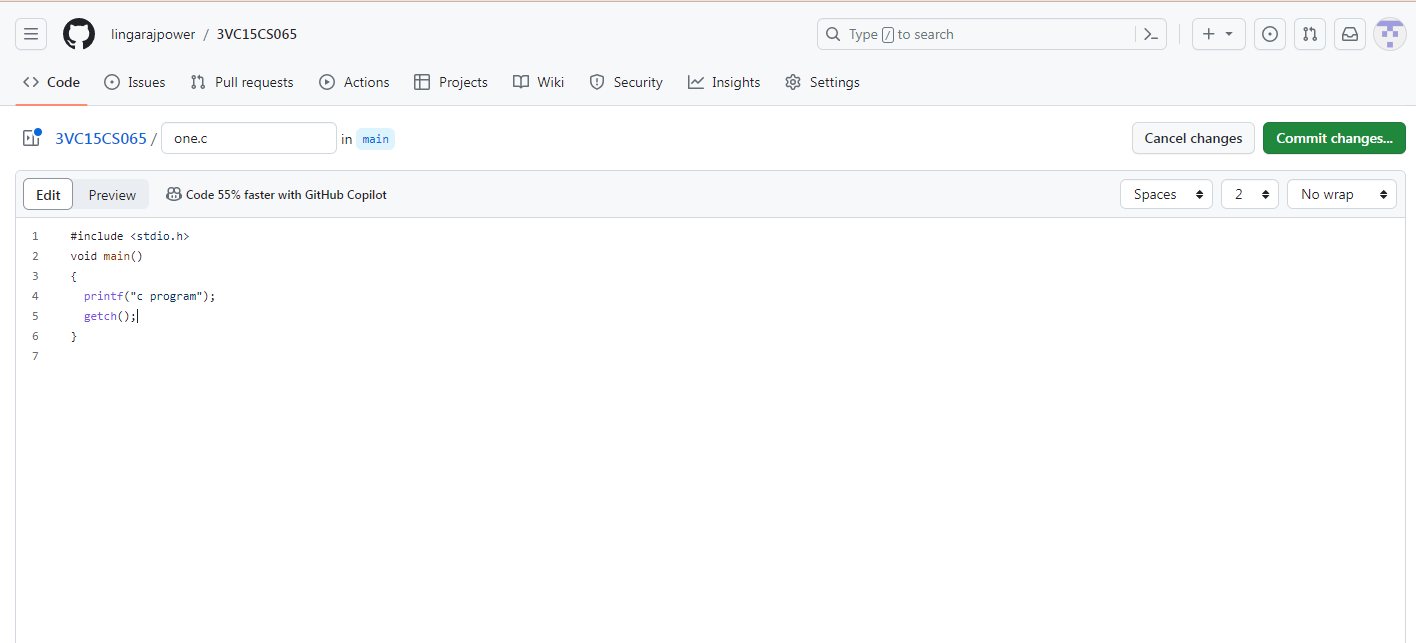


Step 2:

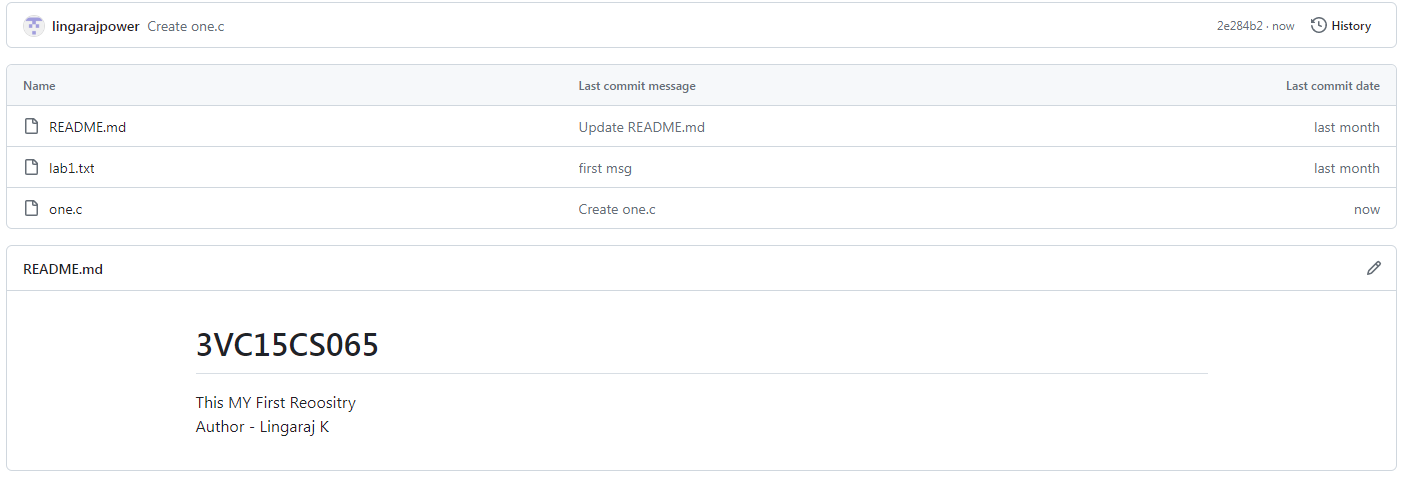


Step 3:

Add file → create new file



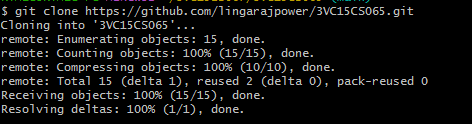
Step 4: commit changes →commit changes



Step 5: go to git bash

Step 6:

Git clone <https://github.com/lingarajpower/3VC15CS065.git>



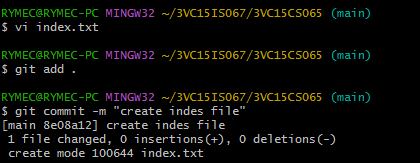
| **Sl. No** | **Experiments** |
| --- | --- |
| 5 | Collaboration and Remote Repositories Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch. |

Step 1:

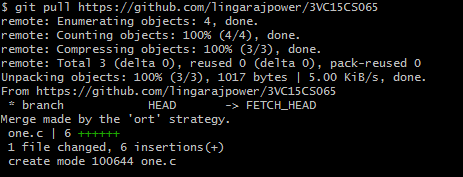
vi index.txt

git add .

$ git commit -m "create indes file"



$ git pull <https://github.com/lingarajpower/3VC15CS065>



ls



Step 2: settings developers setting →personal access tokens→personal acess tokens(classic)→genetrate new token

Step 3: name →mytoken

Expiration → customize to 6 months

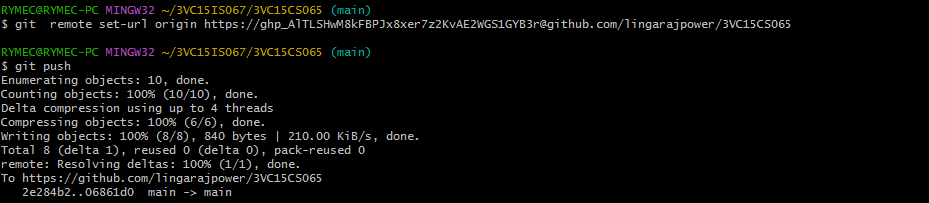
Select all box and submit

Copy token in

Step 4: go to git bash paste

git remote set-url origin <https://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@github.com/lingarajpower/3VC15CS065>

git push



| **Sl. No** | **Experiments** |
| --- | --- |
| 6 | Collaboration and Remote Repositories Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merge. |

Vi pgm6.c

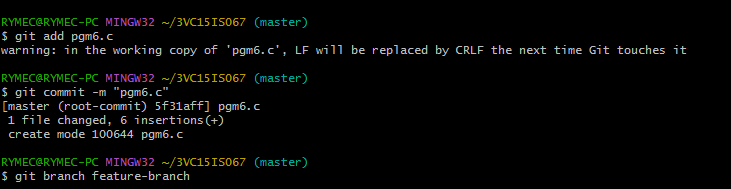


Git add pgm6.c

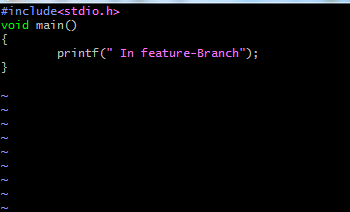
Git commit –m “program 6”

Git branch feature-branch

Git checkout feature-branch

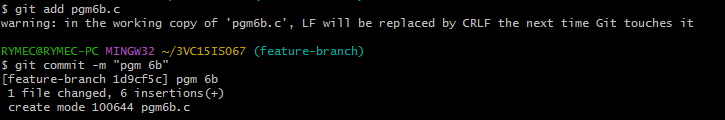


Vi pgm6b.c



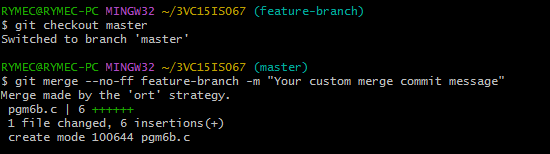
Git add pgm6b.c

Git commit –m “pgm 6b feature branch”



git checkout master

git merge --no-ff feature-branch -m "Your custom merge commit message"



**Note: Here's a breakdown of the command:**

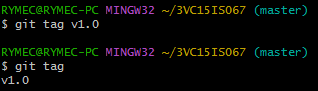
* **git checkout master:** This switches the current working directory to the master branch.
* **git merge --no-ff feature-branch:** This merges feature-branch into the current branch (master). The --no-ff flag prevents git from performing a fast-forward merge. Fast-forward merges can happen when the merging branch (in this case, master) has no commits that the merged branch (feature-branch) does not have. By using --no-ff, it ensures that a new commit is created for the merge, preserving the history of the feature branch as a distinct branch.
* **-m "Your custom merge commit message":** Allows you to provide a custom message for the merge commit. Replace Your custom merge commit message with the actual message you want to use.
* Remember to replace "Your custom merge commit message" with the actual message you wish to use for documenting the merge. This approach ensures the merge is clearly documented and identifiable in the project's history.

| **Sl. No** | **Experiments** |
| --- | --- |
| 7 | Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository. |

git checkout main

git tag v1.0

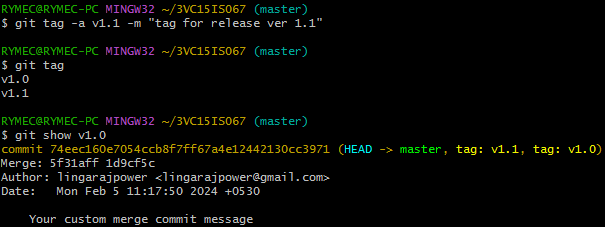
git tag



git tag -a v1.1 -m "tag for release ver 1.1"

git tag

git show v1.0



Git push origin v1.0

| **Sl. No** | **Experiments** |
| --- | --- |
| 11 | Write the command to display the last five commits in the repository's history. |

$ echo "I am in 11a" >> lab11a.txt

$ echo " I am in 11b " >> lab11b.txt

$ echo " I am in 11c " >> lab11c.txt

$ echo " I am in 11d " >> lab11d.txt

$ echo " I am in 11e " >> lab11e.txt

git add .

$ git commit -m "create indes file"

$ git log –n 5

| **Sl. No** | **Experiments** |
| --- | --- |
| 10 | Write the command to list all commits made by the author "user" between "2024-01-01" and "2024-02-07." |

$ echo "I am in 10a" >> lab10a.txt

git add .

$ git commit -m " I am in 10a "

$ echo "I am in 10b" >> lab10b.txt

git add .

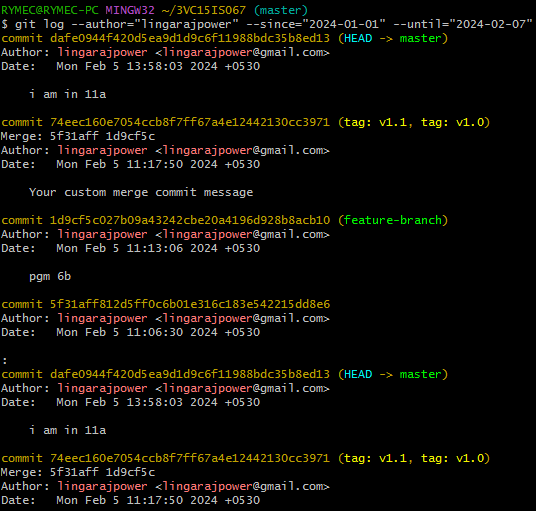
$ git commit -m " I am in 10b "

$ echo "I am in 10c" >> lab10c.txt

git add .

$ git commit -m " I am in 10c "

$ git log --author="lingarajpower" --since="2024-01-01" --until="2024-02-07"



| **Sl. No** | **Experiments** |
| --- | --- |
| 9 | Write the command to cherry-pick a range of commits from "source-branch" to the current branch. |

$ git checkout master

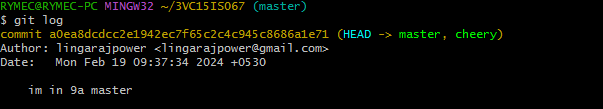
$ echo "I am in 9a" >> lab9a.txt

git add .

$ git commit -m " I am in 9a branch master "

$ git branch cherry

$ git log



$ git checkout cheery

$ echo "I am in 9b" >> lab9b.txt

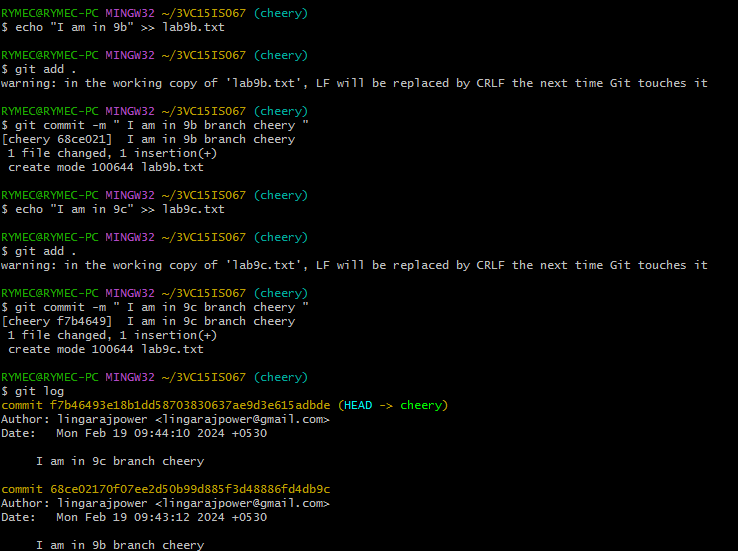
git add .

$ git commit -m " I am in 9b branch cheery "

$ echo "I am in 9c" >> lab9c.txt

git add .

$ git commit -m " I am in 9c branch cheery "

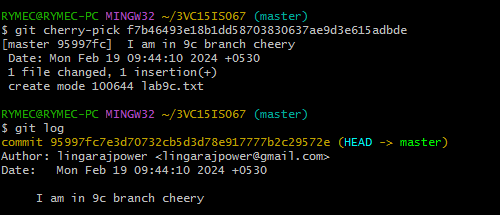


$ git checkout master

$ git log

$ git cherry-pick f7b46493e18b1dd58703830637ae9d3e615adbde

$ git log



| **Sl. No** | **Experiments** |
| --- | --- |
| 8 | Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message? |

$ git checkout master

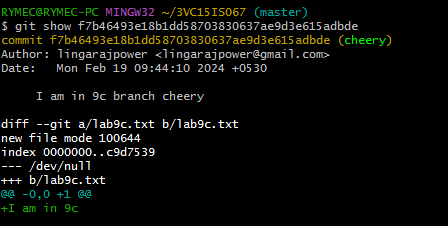
$ echo "I am in 8a" >> lab8a.txt

git add .

$ git commit -m " I am in 8a branch master "

$ git log

$ git show f7b46493e18b1dd58703830637ae9d3e615adbde



| **Sl. No** | **Experiments** |
| --- | --- |
| 12 | Write the command to undo the changes introduced by the commit with the ID "abc123". |

$ git checkout master

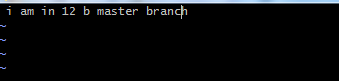
$ vi 12a.txt



git add .

$ git commit -m " I am in 12a branch master "

$ vi 12b.txt



git add .

$ git commit -m " I am in 12b branch master "

$ vi 12a.txt



git add .

$ git commit -m " I am in 12a branch master changes 1"

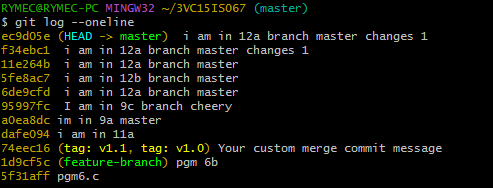
$ vi 12b.txt

.

git add .

$ git commit -m " I am in 12b branch master changes 1 "

$ git log --oneline



$ git revert ec9d05e

